Indoor assistance for visually impaired people using a RGB-D camera

Michiel Vlaminck[1], Hiep Quang Luong[1], Hoang Van Nam[2], Hai Vu[2], Peter Veelaert[1] and Wilfried Philips[1]
[1]Image Processing and Interpretation (IPI), Ghent University, iMinds, Belgium
[2]International Research Institute MICA, Hanoi University of Science and Technology (HUST), Vietnam
Email: michiel.vlaminck@telin.ugent.be

***Abstract*—In this paper a navigational aid for visually impaired people is presented. The system uses a RGB-D camera to perceive the environment and implements self-localization, obstacle detection and obstacle classification. The novelty of this work is threefold. First, self-localization is performed by means of a novel camera tracking approach that uses both depth and color information. Second, to provide the user with semantic information, obstacles are classified as walls, doors, steps and a residual class that covers isolated objects and bumpy parts on the floor. Third, in order to guarantee real time performance, the system is accelerated by offloading parallel operations to the GPU. Experiments demonstrate that the whole system is running at 9 Hz.**

I. Introduction

According to the World Health Organization (WHO), a whopping 90% of the blind people live in developing countries. As they do not have the money to buy advanced assistance solutions, the white cane is still the most used tool although it still has its limitations, e.g. in the case of *aerial* obstacles such as tree branches or fire extinguishers hanging in small corridors. As a result, many systems have been presented in the computer vision literature. The proposed solutions are using a variety of sensors including GPS, laser scanners, stereo cameras and active 3D cameras. Regrettably, GPS can not be used indoor and is not able to provide additional information about the environment. Laser scanners are better suited in this respect, but these scanners are generally too expensive and too impractical to carry around. Nonetheless, in [14] a special system was designed for blind people with walking disabilities where a walker was used to mount two planar laser scanners. Stereo vision is a cheap alternative, but the fact that depth has to be estimated increases the processing time and makes these solutions less robust to illumination changes and shadows. Active 3D camera's are hence the most interesting sensors as they are capable of producing relatively accurate depth maps with sufficiently high resolution at 15 Hz or higher. These sensors either use structured light or are based on the time of flight principle. Undoubtedly one of the most appealing developments in the field of portable 3D cameras is Project Tango, recently announced by Google. The most interesting thing about the Tango tablet is that it incorporates the NVIDIA's Tegra K1 GPU, currently the only mobile GPU that supports CUDA. The development of these devices is attractive since their parallel computation power can make it possible to perform not only the task of self-localization or obstacle detection in real time but go even beyond that by performing additional obstacle classification or semantic annotation of the scene.

II. Related work

Computer vision based assistance devices for the blind generally consist of two parts: 1) the estimation of the ego-motion (i.e. odometry) and 2) the detection of obstacles. The former part is related to simultaneous localization and mapping (SLAM) where the goal is to build a map of the environment while at the same time keep track of the position and orientation of the camera. The spectrum of vision-based SLAM approaches is very broad. A lot of systems rely on a *feature based* framework, characterized by algorithms that track a few features (e.g. SIFT or FAST) in an image sequence, typically in combination with Bayesian filtering to obtain an accurate estimate of the camera pose [1]. Some navigational systems for the blind use a stereo camera that allows to compute 3D points upon which the camera pose is estimated [11], [9], [7]. However, feature-based methods naturally result in sparse representations of the environment and as a consequence they have limited utility in navigational tasks such as obstacle detection and classification. This shortcoming has motivated the development of dense mapping approaches which have been facilitated by the advent of inexpensive RGB-D cameras and parallel computing devices such as the GPU. At the core of these algorithms lies the registration of consecutive point clouds by means of the iterative closest point (ICP) algorithm instead of matching visual features [5]. Since this method only rely on 3D information, it can fail in areas with poor geometrical features and for this reason some researchers tried to combine ICP with visual feature matching [3]. Regarding the actual obstacle detection, a distinction can be made between existing solutions based on whether or not the method is relying on the detection of the ground plane. The former methods mostly use the concept of an occupancy grid, which is a quantization of the space in front of the user into a number of bins. In [10], the authors define a polar grid in front of the user which is subsequently projected on the ground plane and the *bins* of the grid are filled with 3D points that are located within the bounds of each bin. Obstacles are then defined as cells containing more then a predefined number of points. In [6], the approach of the authors differs fundamentally from the previous one in the sense that the algorithm does not start from the detection of the ground plane. Instead, a segmentation is made of the entire scene on pixel level and together with the associated depth values, a decision is made on which objects could possibly act as an obstruction for the user. In this work, we will improve the work of [13], in which a real-time obstacle detection system was presented that uses data captured by RGB-D cameras. We will elaborate on how we can speed up the detection process by offloading parallel operations to the GPU and extend the system by performing real time camera

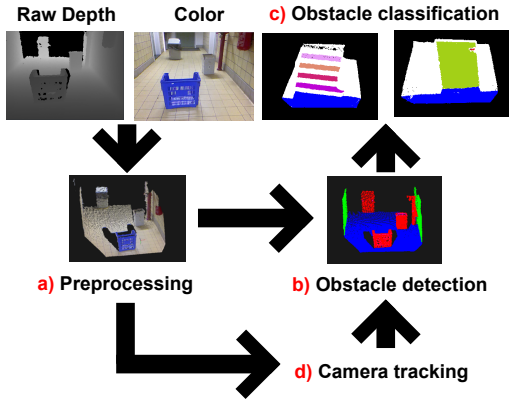

Figure 1. System overview.

tracking in order to obtain ego-motion estimation.

III. Approach

An overview of the system is depicted in figure 1. As mentioned in the introduction it consists of three main parts: self-localization (cfr. ego-motion estimation), obstacle detection and obstacle classification. Self-localization is performed by means of a camera tracking algorithm that uses both depth and color information in an ICP-based algorithm. The result of this camera tracking, i.e. the 6DoF camera pose, is used to build a global 3D map of the environment. Each time new points are *discovered*, we fuse them into this global map, which is based on the octree data structure. We are using the Octomap framework presented in [4] but extended it to keep semantic information on the type of possible obstacles such as walls, doors and steps. This global 3D map of the environment can be used to (re)locate the user in case the ego-motion algorithm has failed. We can also use it to determine if obstacles are moving in the scene. The obstacle detection itself is based on the segmentation of the *current* scene in 3D. Using the segmentation of the scene, we make a decision on which objects could possibly act as an obstruction for the user. The obstacle classification, finally, is led by the geometrical properties of respectively a wall, a staircase or a door. In order to keep the total operation time as low as possible, we offload as many parallel operations as possible to the GPU. To this end, the system was partly implemented using Quasar, a heterogeneous programming framework [2] which facilitates parallel programming on the GPU.

A. Preprocessing

In the preprocessing phase, we convert the incoming depth and color data prior to analysis. The first step is to convert the depth map to an *organized* point cloud $\mathbf{P}$ containing 3D points. Organized denotes that the point cloud still has its projection in 2D and thus has a fixed width and height. This task can be done in parallel by letting each CUDA thread operate on a separate pixel in the incoming depth map $\mathbf{D}$. The second step consists of computing surface normals for each point leading to a single normal map $\mathbf{N}$. Using the 2D projection of the point cloud $\mathbf{P}$, this can be computed extremely fast by considering two vectors that are tangential to the local surface at point $\mathbf{p}$. These vectors can be computed using the left, right, upper and lower neighbouring point in $\mathbf{P}$. The normal can then be obtained by taking their cross product. In the final step, we transform the point cloud in order to align the ground plane with the xz-plane in our reference system using data captured by an accelerometer. This transformation helps the camera tracking process in providing an initial guess for the orientation of the camera. In addition, it simplifies the segmentation of the ground plane since we can rely on the fact that the ground plane has a normal perpendicular to the xz-plane.

B. Obstacle detection

The first step in the obstacle detection process is to segment the environment into regions that geometrically belong to each other. An indoor scene will typically consist of large planes (e.g. floor, walls, ceiling, ...) and for that reason, the segmentation process will first try to identify these planes. To this end, we exploit the organized nature of the point clouds by scanning it in the image domain thereby comparing neighbouring pixels in a 4-connected way. Each pixel is assigned a label according to its properties compared to its neighbours. More precisely, we use two different comparison functions, the first one testing for the euclidean distance of the corresponding 3D points, $||\mathbf{p}_i - \mathbf{p}_{i-1}||_2 < \epsilon$, the second one testing for the deviation in normal vector $\mathbf{n}_i \cdot \mathbf{n}_{i-1} < \cos\theta$. Only when these two conditions are met, the two neighbouring pixels are assigned the same label. To obtain the label images we first compute a binary image (i.e. mask), which we then feed to our own GPU-based connected component labeling (CCL) algorithm. Once the dominant planes in the scene are segmented, the remaining objects are often easily distinguishable as seen on the middle image of figure 2. It is thus easy to segment these objects by clustering the points based on their relative distance. To this end, we will again use CCL, but this time only using the condition on euclidean distance. The result of this entire process can be seen in the right image of figure 2.

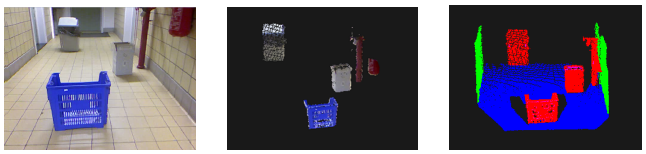

Figure 2. Remaining objects after the segmentation of the dominant planes (middle). Result of euclidian clustering by means of CCL (right).

C. Obstacle classification

Once the scene is segmented in different planes and objects, the next step is to classify them. We consider four different classes, including staircases, doors, walls and a residual class that covers loose obstacles and bumpy parts on the floor. A staircase can be modelled as a set of planes (i.e. steps) that are all parallel to the ground plane and that are located at the same distance from each other. This is a simple but quite strong model as it can also recognize the challenging case of spiral staircases. The first step in the process is to test the previously segmented planes based on their normal vector, which should indeed be pointing in the direction of the y-axis. This gives a set of candidates steps S. Subsequently, we look for the plane closest to the ground plane and test if it lies below the maximum height of 30 cm. If such a plane is found, we determine if its number of inliers is higher than a certain threshold. This threshold is variable as steps further away will contain less points than staircases that are close by. Once such a plane is found, we conclude that we are dealing with a step and we repeat the process, looking for another

plane within a distance of 30 cm from the previous one. The process stops when no more planes are found. Note that we test both, for ascending and descending steps. Doors on the other hand can be defined as a plane perpendicular to the ground plane having a certain width and a door handle at a certain position. However, the plane of the door is often not distinguishable from the plane representing the wall. Figure 3 depicts two example doors where the first one represents a so called *non-concave* door whereas the second one represents a *concave* one. In the first case, our previously described plane segmentation algorithm will segment the wall and door as one single plane.

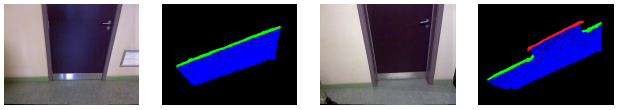

Figure 3. Example of a *non-concave* door (left) and a *concave* door (right).

The solution for this problem is to incorporate color information, as the color of the door often differs from the color of the wall, cfr. figure 3. We will still adopt our CCL algorithm but extend it with an additional comparison function based on the registered color image associated with the point cloud. The additional comparison function is given by $||\mathbf{c}_i - \mathbf{c}_{i-1}||_1 < \epsilon$, in which $\mathbf{c}_{i-1}$ and $\mathbf{c}_i$ are the color vectors of two neighbouring pixels. Each of the components (h, s, v in case of HSV colour space) are subtracted and finally summed. Using this additional condition, we can segment each earlier segmented plane in different sub planes. The final step then consists of testing the width of the obtained planes and looking for the door handle. To define the width of the door, we first compute the convex hull of the set of points belonging to the plane using the quickhull algorithm. Only when the plane fulfils the constraint on the width and when a door handle is present, we will conclude that we are dealing with a door.

D. Camera tracking

The last part of our system consists of tracking the camera pose of each consecutive frame. In the following sections we will discuss the main steps of our ICP-based approach.

1. Point Selection The first step is to select the points which we will use to compute the transformation. Hereby, it is important that we only select points that do not belong to moving objects. Since we already made a segmentation of the scene in the obstacle detection process we can exploit this knowledge to determine if an object is moving by using the geometric consistency property. The main idea of this approach is to verify if all the segments have consistent relative locations in two consecutive frames. In other words, the distance between two objects should remain constant throughout the sequence. To this end we first perform a bipartite assignment for each segment, i.e. an association between the segments of one frame with the ones of the next frame, which we obtain using the Hungarian algorithm. Next, we test for each possible pair of associations the distances between the centres of mass in the current frame (d_c) and previous frame (d_p). If the difference between d_c and d_p is close to zero, i.e. below a certain threshold, the positions of the two centres of mass in each frame is considered consistent. For each object, we keep track of the number of other objects that are geometrical consistent. Only the objects with a count sufficiently high are considered as static objects, the other ones are considered as moving objects. The points lying in the segments corresponding to a moving object are removed from the selection process.

2. Point Matching The next step is to determine the correspondence between points in two consecutive clouds. In [5], this is achieved by projective data association (PDA). In PDA, the corresponding point of a point $\mathbf{s_i} \in \mathbf{S}$ is the closest point $\mathbf{d_i} \in \mathbf{D}$ to the line of projection of $\mathbf{s_i}$ into the camera used to obtain the points of the second cloud $\mathbf{D}$. Point correspondences are thus computed by projecting one point cloud onto the other with respect to the other's camera. However, PDA has the limitation that it fails in areas that are poor in geometry. In our case however, it will frequently happen that a user is walking through a long hallway or corridor. Since these corridors mainly consist of a floor and two flat walls, the scene captured by the camera lacks geometric features. A second limitation of PDA is that it introduces a limit in terms of the maximum rotation and translation between two consecutive frames that still allows reasonable data association. To overcome these limitations, we decided to use color information in order to find point correspondences. Instead of using PDA, we perform data association by generating a corresponding map C_{map} based on the flow field found by the Horn-Schunck algorithm. We implemented our own GPU-based method, which is based on the multi-scale strategy described in [8]. The main advantages of our approach over other methods that take color information into account is that there is no need for expensive feature point matching and the fact that *all* color information is incorporated.

3. Rejection For a more accurate estimate of the camera pose, corresponding pairs computed in the previous section are only considered when a correct alignment is guaranteed. Let us therefore define ϵ as the threshold for the maximum distance of a corresponding pair and θ as the difference in normal vectors. A corresponding pair $(\mathbf{s_i}, \mathbf{d_i})$ is rejected if either one of the following conditions $||\mathbf{R}\mathbf{s_i} + \mathbf{t} - \mathbf{d_i}||_2 > \epsilon$ or $\mathbf{R}\mathbf{n}'_\mathbf{i}(\mathbf{n_i})^\top > \theta$ is met. Herein $\mathbf{R}$ is the estimated rotation, $\mathbf{t}$ the estimated translation and $\mathbf{n}'_\mathbf{i}$ the normal at the source point.

4. Transformation estimation Once the point correspondences are found, we can compute the actual transformation that aligns the two point clouds. In traditional ICP approaches, the most commonly used error metric is the sum of the euclidean distances between corresponding points after alignment. However, we adopt a point-to-plane error metric as this has been shown to converge much faster. The object of minimization in this error metric is the sum of the squared distance between a point and the tangent plane of its correspondence point given by $E(\mathbf{S}, \mathbf{D}; \mathbf{T}) = \sum_{i=1}^{N} ||(\mathbf{T}\mathbf{s_i} - \mathbf{d_i}) \cdot \mathbf{n_i}||^2$. In this equation $\mathbf{n_i}$ is the surface normal at $\mathbf{d_i}$, which we already computed in parallel for each point of $\mathbf{D}$. The final transformation matrix is then given by $\mathbf{T} = \underset{\mathbf{T}}{\operatorname{argmin}}\, E(\mathbf{S}, \mathbf{D}; \mathbf{T})$.

IV. Evaluation

Our entire system was evaluated on a notebook with a NVIDIA GeForce GT 750M and an Intel Core i7 4700HQ processor inside. We can report that the preprocessing, including the conversion from depth map to point cloud and the normal estimation, takes approximately 10 ms. Regarding the obstacle detection process, the segmentation of the scene for images with a resolution of 640×480 takes approximately 20 ms. The obstacle classification part hardly needs any additional

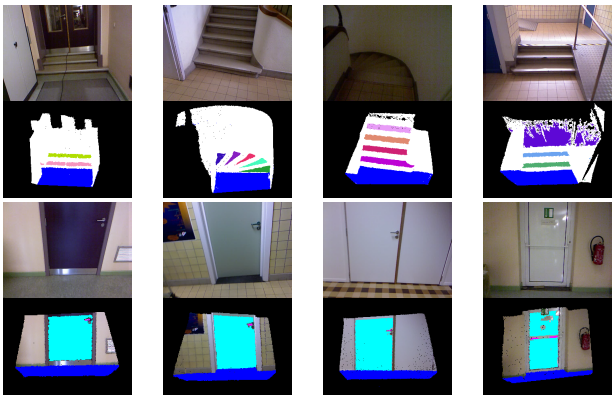

Figure 4. Examples of detected staircases and doors. Each of these images are selected from sequences recorded at the Technicum building of Ghent University.

time to detect staircases as this process mainly relies on the previous segmentation. Identifying doors takes an additional 20 ms due to the computation of the convex hulls of the planes to determine their absolute width. Finally, the camera tracking takes approximately 60 ms per frame. The selection and matching of points is very fast, but the transformation estimation is rather slow due to the linear system that is solved on the CPU. All together, the entire system needs approximately 110 milliseconds to process each new frame. Regarding robustness, the most critical part of our system is the identification of the ground plane. If the ground plane is wrongly estimated, the entire obstacle detection process will fail. Fortunately, when the ground plane is not visible in the image, we can still rely on the output of the camera tracking to locate it. If the ground plane is correctly identified, the detection of obstacles is very reliable. In order to evaluate the reliability of the step detection process, we used the dataset that was collected by Tang *et al.* in [12]. This dataset consists of 90 positive images of staircases together with their corresponding depth map and accelerometer data. In addition, a negative collection of 58 entries is provided. Regarding the 90 positive test images, we can report that no stairs remained undetected by our method. Moreover, our method lead to only 2 false positives both of them due to the fact that the ground plane was hardly visible in the image. Besides the dataset presented in [12], we also collected our own dataset containing images and corresponding depth maps of 27 different staircases. Each of these images are selected from sequences that we recorded at the Technicum building of Ghent University. As can be seen in figure 4, our method also works for spiral staircases and in bad lighting conditions which makes it a lot more robust compared to existing systems that are solely based on images. Our algorithm was able to correctly detect all of the 27 different staircases. Regarding door detection, we selected another 44 images from the same sequences. Figure 4 shows that even tricky cases such as dual doors or emergency doors were correctly detected. Only 6 out of 44 doors remained undetected. The main reasons for these failures are the lack of sufficient ambient light and the fact that the door handle was not correctly detected due to the combination of noise and its thin structure.

V. Conclusion

In this paper a novel assistance aid for the visually impaired was presented. The system is unique in different ways. First, we combine self localization with obstacle detection and obstacle classification. Second, we exploit the parallel computing power of the GPU to ensure that both tasks can be executed in nearly real time. Third, we extended existing camera tracking algorithms by combining both depth and color information. Our method differs from existing methods in the sense that we do not adopt computational expensive feature extraction but instead incorporate optical flow as a data association technique.

Acknowledgement

This research is co-funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number FWO.102.2013.08.

References

[1] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007. 1

[2] B. Goossens, J. De Vylder, and W. Philips. Quasar - A new heterogeneous programming framework for image and video processing algorithms on CPU and GPU. In *2014 IEEE International Conference on Image Processing, ICIP 2014, Paris, France, October 27-30, 2014*, pages 2183–2185, 2014. 2

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *Int. J. Rob. Res.*, 31(5):647–663, April 2012. 1

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at http://octomap.github.com. 2

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, pages 559–568, 2011. 1, 3

[6] Chia-Hsiang Lee, Yu-Chi Su, and Liang-Gee Chen. An intelligent depth-based obstacle detection system for visually-impaired aid applications. In *WIAMIS*, pages 1–4. IEEE, 2012. 1

[7] Tung-Sing Leung and Gérard G. Medioni. Visual navigation aid for the blind in dynamic environments. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 579–586, 2014. 1

[8] E. Meinhardt-Llopis, J. Snchez Prez, and D. Kondermann. Horn-Schunck Optical Flow with a Multi-Scale Strategy. *Image Processing On Line*, 3:151–172, 2013. 3

[9] V. Pradeep, G. Medioni, and J. Weiland. Robot vision for the visually impaired. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 15–22, June 2010. 1

[10] A. Rodrguez, J. J. Yebes, P. F. Alcantarilla, L. M. Bergasa, J. Almazn, and A. Cela. Assisting the visually impaired: Obstacle detection and warning system by acoustic feedback. *Sensors*, 12(12):17476, 2012. 1

[11] J. M. Saez, F. Escolano, and A. Penalver. First steps towards stereo-based 6DOF SLAM for the visually impaired. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 0:23, 2005. 1

[12] T.J.J. Tang, W.L.D. Lui, and W.H. Li. Plane-based detection of staircases using inverse depth. In *Australasian Conference on Robotics and Automation*, Wellington, New Zealand, 2012. 4

[13] M. Vlaminck, L. Jovanov, P. Van Hese, B. Goossens, W. Philips, and A. Pizurica. Obstacle detection for pedestrians with a visual impairment based on 3d imaging. In *2013 International Conference on 3D Imaging, Proceedings*, page 6. IEEE, December 2013. 1

[14] A. Wachaja, P. Agarwal, M. Reyes Adame, K. Möller, and W. Burgard. A navigation aid for blind people with walking disabilities. In *IROS Workshop on Rehabilitation and Assistive Robotics: Bridging the Gap Between Clinicians and Roboticists*, Chicago, USA, 2014. 1